Data Science Methods

Homework Assignment 1

Mike Weltevrede SNR: 1257560
Robbie Reyerse SNR: 2039047
Steffie van Poppel SNR: 2031218

21 February 2020

1 Principal Component Analysis

We begin by presenting the code for importing and preprocessing the data.

```
rm(list = ls())

library(reshape2)
library(ggplot2)

readdata.function <- function(p) {
  library("readxl")
  all_data <- readxl::read_excel(p)
  airpol <- all_data[which(all_data[,1] == "AIRPOL"), 2]
  airpol <- airpol[[1]]

  return(list(all_data = all_data, airpol = airpol))
}

PCA_dataprep.function <- function(AIRPOL, all_data) {
  selected_data_00 <- all_data[which(all_data[, 2] == AIRPOL):nrow(all_data),
      ]
  if (length(which(is.na(selected_data_00[, 1]))) == 1) {
    selected_data_01 <- t(selected_data_00[
      (which(is.na(selected_data_00[, 1])) + 1):nrow(selected_data_00), ])
  } else {
    selected_data_01 <- t(selected_data_00[
      (which(is.na(selected_data_00[, 1]))[1] + 1):(which(
        is.na(selected_data_00[, 1]))[2] - 1), ])
  }

  selected_data_02 <- selected_data_01[2:nrow(selected_data_01),
                                       2:ncol(selected_data_01)]
  rownames(selected_data_02) <- selected_data_01[2:nrow(selected_data_01), 1]
  colnames(selected_data_02) <- selected_data_01[1, 2:ncol(selected_data_01)]
  class(selected_data_02) <- 'numeric'
  scaled_data <- scale(selected_data_02)

  return(list(scaled_data = scaled_data, selected_data_02 = selected_data_02))
}

# Get data
airpol <- readdata.function(p = "data\\env_air_emis.xls")
data_sul <- PCA_dataprep.function(airpol$airpol[5], airpol$all_data)
```

1.1

We want to run principal component analysis (PCA) on standardised data. For this PCA, we are interested in the first two PCs. Figure 1 displays the data points along with the loadings for the first two principal components. For PC1, most countries have the same correlation towards the principal components seeing as the direction of the loading arrows are quite similar and negative. However, it is noticeable that a few countries (Malta, Cyprus, and Greece) are explained less in PC1 than the rest as their PC1 coordinate is closer to 0. This indicates that this PC does not really measure the level of air pollution of this area. The opposite holds for PC2. Here these three countries are the countries with the highest loading, thus this PC does mainly measure the level of air pollution for these areas, while the PC2 coordinate for the other countries is not that large. However, we do not see any statistical information about the loadings, e.g. standard deviations. This means that when the analysis would be repeated with a new sample we do not know in what range the arrows could change. Therefore, we cannot directly make conclusions on the correlations between the variables using the absolute size of the loadings.

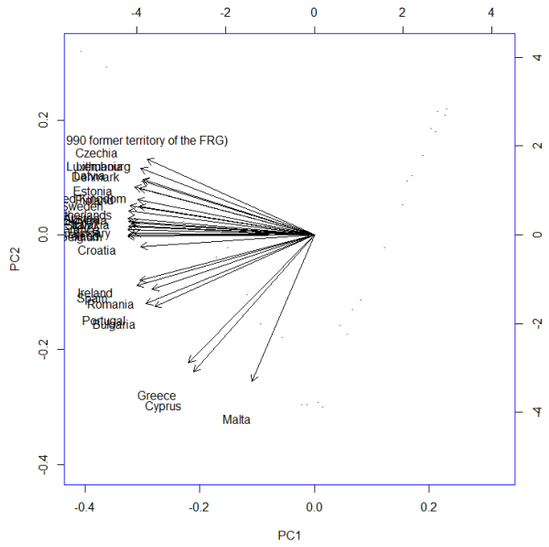

Figure 1: First two principal components

The purpose of applying principal component analysis is dimension reduction. That is, we have many independent variables (or features) and we want to capture (nearly) the same information that these give us in fewer sources of information. PCA tries to reduce the number of features from p to some lower number k. It does this by identifying linear combinations of variables along which the data varies the most. These linear combinations are the principal components. These are independent of each other by construction since PCA imposes an orthogonality assumption when calculating the PCs and their loadings. This all implies that we can capture the variance of many dependent variables in a much smaller set of variables.

Do note that one should be very careful interpreting the values of the loadings. Firstly since PCA is generally applied to scaled data. Scaling is necessary because, otherwise, big countries like Germany are compared with absolute numbers to far smaller countries like Luxembourg, meaning that the possible effect of Luxembourg bleakens when compared to Germany. We should also be careful because we do not have standard deviations available, meaning that we cannot test values comparatively. For example, suppose that for one PC we have loadings 0.55, 0.01, and 0.17. We cannot definitively state that this PC therefore mostly shows the effect for the first category as we cannot prove that 0.55 is actually statistically significantly larger than 0.17, for instance.

1.2

The code below generates screeplots of the eigenvalues ordered from largest to smallest in two ways: the proportion of the variance that is explained by all principal components, as well as the cumulative variance.

```
# PCA plots (Q1 - a and b):
PCA_sul <- prcomp(data_sul$scaled_data)
load_PC1PC2 <- PCA_sul$rotation[, 1:2]

biplot(PCA_sul, xlabs = rep(".", nrow(PCA_sul$x)), col = c("blue", "black"))

cum_var_per <- cumsum(PCA_sul$sdev^2) / sum(PCA_sul$sdev^2)
var_per <- PCA_sul$sdev^2 / sum(PCA_sul$sdev^2)

par(mfrow = c(1, 2))
plot(var_per, type = 'b', xlab = 'Principal Component',
     ylab = 'Prop. Variance Explained')
plot(cum_var_per, type = 'b', xlab = 'Principal Component',
     ylab = 'Cumulative Prop. Variance Explained')
par(mfrow = c(1, 1))
```

Figure 2 shows the additional proportional variance explained per principal component (left) and the cumulative proportional variance explained by the principal components (right). As a rule of thumb, it can be said that we choose the lowest amount of principal components such that the cumulative proportion of variance explained is at least 80%. For this data, this is already obtained when using

only one principal component. For the Sulphur pollutant data, it would therefore be sufficient to only take the first PC.

We can also use the method that we take the number of principal components where we can see the screeplot levelling out (we see an "elbow"). Specifically, we consider the left plot. Based on this method, we would choose perhaps 2 or 3 principal components. This directly indicates the issue: it is not a robust statistical method. Firstly because, as in questions 1.1, the results may vary for a new run, but also because it is arbitrary to indicate where the "elbow" is, especially when the plots become even less clearly indicative. We do not have the standard deviations available, as was the case in the previous question. If these standard deviations would be large, this could make a whole new graph. For example, the proportional variance explained by the first principal component could drop below 80%. We could therefore choose to also choose the first two principal components, or even more.

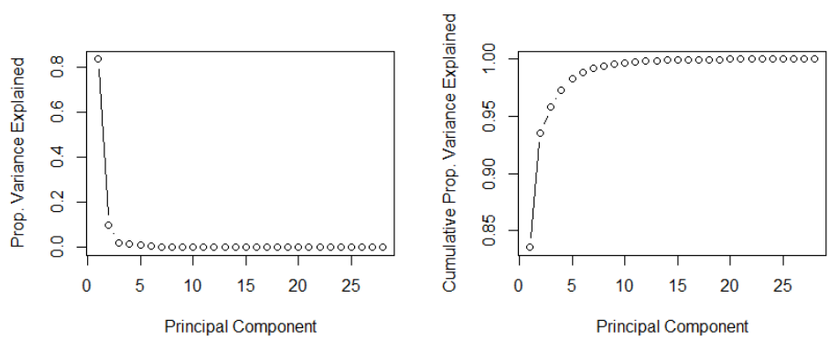

Figure 2: First two principal components, left: additional variation explained per principal component, right: cumulative variation explained over the principal components

1.3

Instead of using the screeplots to select the number of PCs k, we want to use the Bayesian information criterion (BIC) which is given by:

$$BIC(k) = \log(SSR(k)) + \frac{\log(C_{np})}{C_{np}}k \quad k = 1, 2, 3...$$

where $C_{np} = np$ since n and p are equal in our case. In general, $C_{np} = min(n, p)$ can be used (Bai and Ng, 2002). Moreover, we have

$$SSR(k) = \frac{1}{np}\sum_{i=1}^{n}\sum_{j=1}^{n}\hat{\epsilon}_{ij}^2(k) \quad i = 1, ..., n, \ j = 1, ..., p$$

Given the factor model:

$$x_{ij} = \hat{\lambda}_j\hat{f}_i + \hat{\epsilon}_{ij}(k)$$

the residuals are obtained in the following way:

$$\hat{\epsilon}_{ij}(k) = x_{ij} - \hat{\lambda}_j\hat{f}_i$$

where, x_{ij} is the scaled data, $\hat{\lambda}_j$ are the loadings of the first k principal components for country j and $\hat{f}_i$ are the factors (AKA scores when considering PCA) of the first k principal components for year i. Note that PCA itself is not a model, it is a solution method to a factor model, as described above, of which we can take the outcome to determine the residuals.

The SSR will, of course, decrease when including more principal components (k increases), since we will explain more variance and, therefore, the error will decrease. However, the second term of the BIC, the penalty function, will increase linearly by including more principal components. This behaviour can indeed be seen in Figure 3.

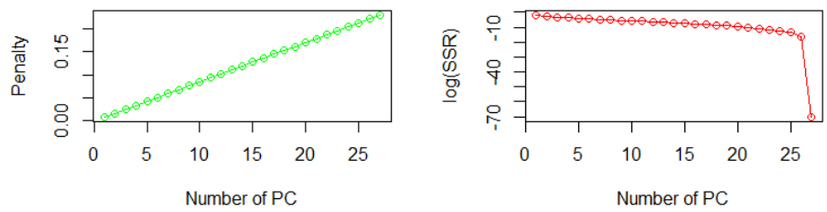

Figure 3: Left: penalty over number of principal components, right: log of the sum of squared residuals over the number of principal components

According to the BIC method, we should choose k such that $BIC(k)$ is minimised. In Figure 4, we plot the BIC over the number of principal components. We can see that the BIC is actually almost fully determined by the log of the SSR and that the values of the penalty are small in absolute terms compared to the very negative values of $\log(SSR)$. Trusting this graph would lead to choosing 27 principal components to be optimal amount (note that we can only go up to $p-1$ PCs here). This seems rather unrealistic as this does not reduce our dimension much.

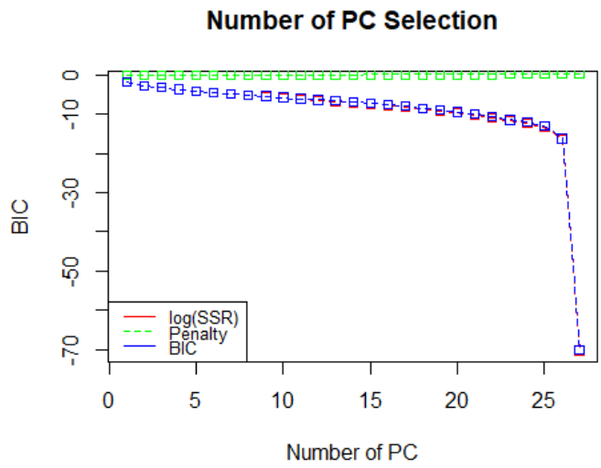

Figure 4: Bayesian information criterion over the number of principal components

Figures 3 and 4 were constructed with the following R code:

```r
# BIC Criteria (Q1c)
if (nrow(data_sul$scaled_data) == ncol(data_sul$scaled_data)) {
  Cnp <- nrow(data_sul$scaled_data)*ncol(data_sul$scaled_data)
} else {
  Cnp <- min(nrow(data_sul$scaled_data), ncol(data_sul$scaled_data))
}

k_max <- ncol(PCA_sul$rotation) - 1

SSR <- rep(0, k_max)
penalty <- rep(0, k_max)
BIC <- rep(0, k_max)

for (k in 1:k_max) {
  epsilon2 <- matrix(0, nrow(PCA_sul$x), nrow(PCA_sul$rotation))

  for (i in 1:nrow(PCA_sul$x)) {
    for (j in 1:nrow(PCA_sul$rotation)) {
      epsilon2[i,j] <- (data_sul$scaled_data[i, j] - PCA_sul$rotation[j, 1:k]
                        %*% t(PCA_sul$x)[1:k, i])^2
    }
  }

  SSR[k] <- sum(epsilon2) / (nrow(PCA_sul$x)*nrow(PCA_sul$rotation))
  penalty[k] <- k*log(Cnp) / Cnp

  BIC[k] <- penalty[k] + log(SSR[k])
```

```
}

par(mfrow = c(1, 2))
plot(penalty, type = "o", col = "green", ann = FALSE)
title(xlab = 'Number of PC', ylab = 'Penalty')
plot(log(SSR), type = "o", col = "red", ann = FALSE)
title(xlab = 'Number of PC', ylab = 'log(SSR)')

par(mfrow = c(1, 1))
plot(log(SSR), type = "o", pch = 22, lty = 2, col = "red",  ann = FALSE)
lines(penalty, type = "o", pch = 22, lty = 2, col = "green")
lines(BIC, type = "o", pch = 22, lty = 2, col = "blue")
title(main = 'Number of PC Selection', xlab = 'Number of PC', ylab = 'BIC')
legend("bottomleft", c("log(SSR)", "Penalty", "BIC"), cex = 0.8,
       col = c("red", "green", "blue"), lty = 1:2)
```

The features x_{ij} that we used were scaled (dividing by all values by the standard deviation) and centered (subtracting all values by the mean), as we noted before. Due to this, the residuals can become very small. Not scaling the data will lead to graph in Figure 5. Because the penalty and the $\log(SSR)$ are not out of proportion, we do get reasonable output; a minimum can be found at $k = 2$. Even though this supports the thought that here the BIC on scaled data does not lead to the right conclusions about the number of principal components that should be used, not scaling the data is also not the right way to tackle the issue. As already explained, the data just consists out of tonnes of pollution of certain pollutants. Not scaling the data will give unrealistic view, since some countries are far bigger, have far more inhabitants than others, and, as a result, likely emit more pollutants.

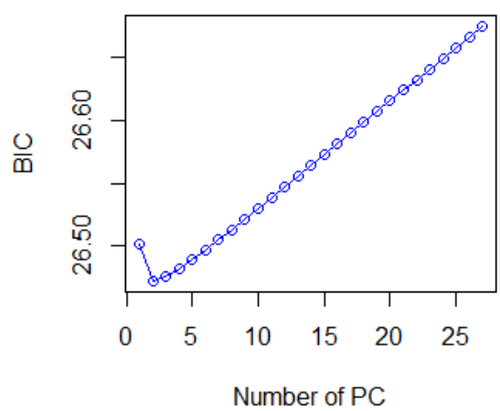

Figure 5: BIC with unscaled data

The unscaled version of the BIC analysis was performed with the following R code:

```
# BIC Criteria (Q1c) - unscaled data
PCA_sul <- prcomp(data_sul$selected_data_02)

if (nrow(data_sul$selected_data_02) == ncol(data_sul$selected_data_02)) {
  Cnp <- nrow(data_sul$selected_data_02)*ncol(data_sul$selected_data_02)
} else {
  Cnp <- min(nrow(data_sul$selected_data_02), ncol(data_sul$selected_data_02))
}

k_max <- ncol(PCA_sul$rotation) - 1

BIC <- rep(0, k_max)

for (k in 1:k_max) {
  epsilon2 <- matrix(0, nrow(PCA_sul$x), nrow(PCA_sul$rotation))

  for (i in 1:nrow(PCA_sul$x)) {
    for (j in 1:nrow(PCA_sul$rotation)) {
      epsilon2[i,j] <- (data_sul$scaled_data[i, j] - PCA_sul$rotation[j, 1:k]
                        %*% t(PCA_sul$x)[1:k, i])^2
    }
  }
```

```
23
24    SSR <- sum(epsilon2) / (nrow(PCA_sul$x)*nrow(PCA_sul$rotation))
25    penalty <- k*log(Cnp) / Cnp
26
27    BIC[k] <- log(SSR) + penalty
28  }
29
30  par(mfrow = c(1, 1))
31  plot(BIC, type = "o", col = "blue", ann = FALSE)
32  title(xlab = 'Number of PCs', ylab = 'BIC')
```

1.4

We now analyse each type of pollutant, 5 in total, over time. We are particularly interested in the trend (dis)similarities between the different pollutants between the first and second principal component. Please consider Figures 6 and 7.

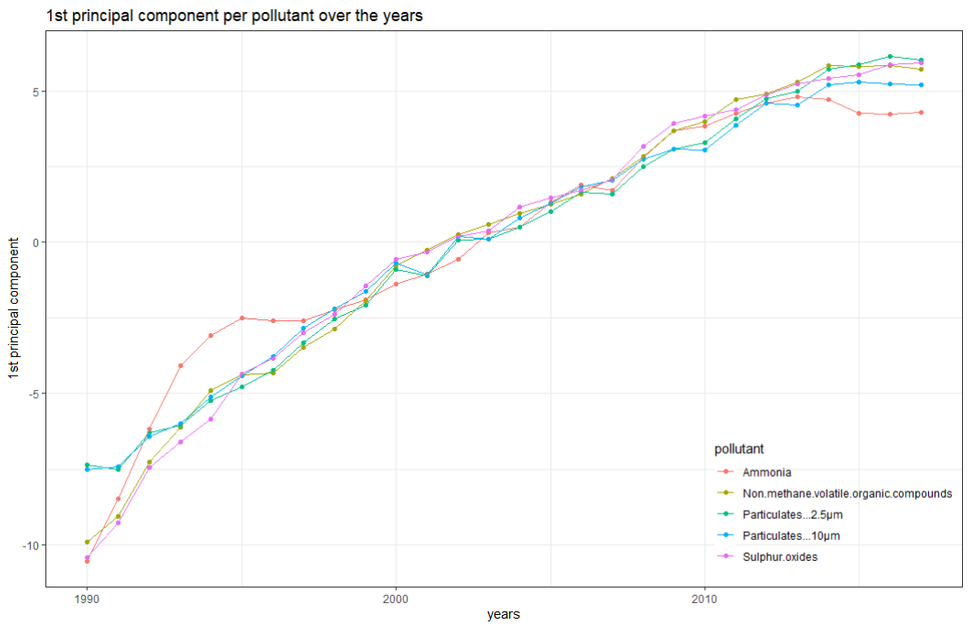

Figure 6: first principal component over the time

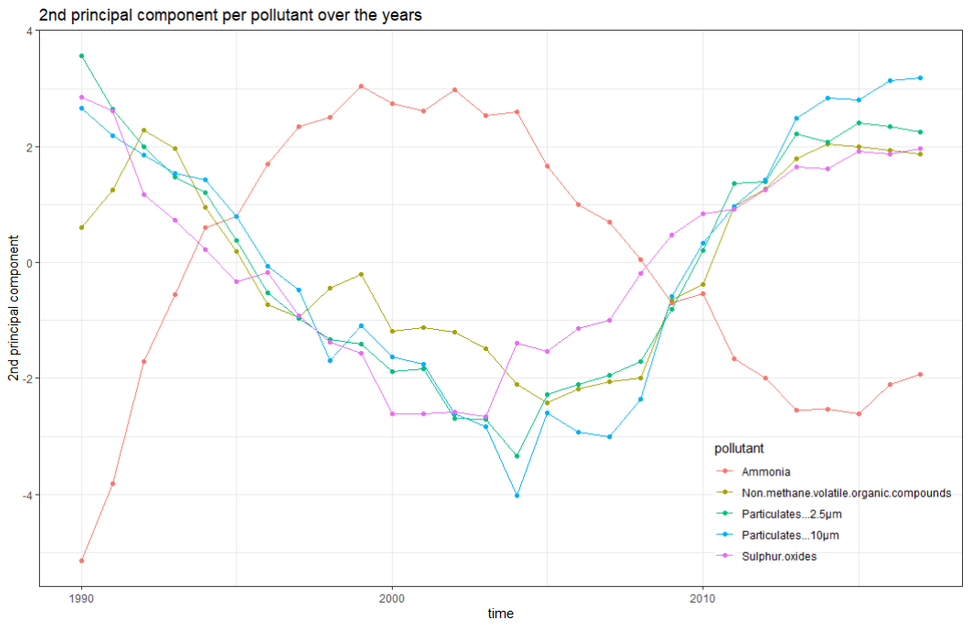

Figure 7: second principal component over the time

We can see that the behaviour of the pollutants is quite similar for the first PC. As such, this PC likely does not show information that is strongly indicative for one pollutant over others. For the second PC we see that almost all pollutants behave similarly. However, Ammonia moves in the exact opposite

direction as the other pollutants for the second PC over time. The conclusion that we can (carefully) draw here, is that the second PC gives us a lot of information about the effects of ammonia. However, we should, of course, be careful with interpreting the exact magnitudes.

The code used to generate these plots is as follows:

```
# Principal components over time (Q1d)
PC1 <- matrix(0, 28, length(airpol$airpol))
PC2 <- matrix(0, 28, length(airpol$airpol))

for (i in 1:length(airpol$airpol)) {
  data <- PCA_dataprep.function(AIRPOL = airpol$airpol[i],
                                all_data = airpol$all_data)
  PCA <- prcomp(data$scaled_data)
  PC1[, i] <- PCA$x[, 1]
  PC2[, i] <- PCA$x[, 2]
}

years <- unname(rownames(PCA$x))

#Principal Component 1 (Q1d)
rownames(PC1) <- rownames(PCA$x)
colnames(PC1) <- airpol$airpol
data_PC1 <- data.frame(PC1,years)
data_PC1 <- reshape2::melt(data_PC1, id.vars = 'years')
data_PC1$years <- as.numeric(as.character(data_PC1$years))
colnames(data_PC1)[3] <- 'pollution'
colnames(data_PC1)[2] <- 'pollutant'

ggplot(data_PC1, aes(years, pollution, col = pollutant)) +
  geom_point() + geom_line() + ylab('1st principal component') +
  ggtitle("1st principal component per pollutant over the years" ) +
  theme_bw() + theme(legend.position = c(0.86, 0.15),
                     legend.background = element_blank())

#Principal Compent 2 (Q1d)
rownames(PC2) <- rownames(PCA$x)
colnames(PC2) <- airpol$airpol
data_PC2 <- data.frame(PC2,years)
data_PC2 <- melt(data_PC2, id.vars = 'years')
data_PC2$time <- as.numeric(as.character(data_PC2$years))
colnames(data_PC2)[3] <- 'pollution'
colnames(data_PC2)[2] <- 'pollutant'

ggplot(data_PC2, aes(time, pollution, col = pollutant)) +
 geom_line() + geom_point() + ylab('2nd principal component') +
  ggtitle("2nd principal component per pollutant over the years" ) +
  theme_bw() + theme(legend.position = c(0.86, 0.15),
                     legend.background = element_blank())
```

2 NIPALS

To explain why the NIPALS algorithm consistently estimates the PC solution to a factor model we will first explain the intuition behind NIPALS. For this, it will be assumed that $p < n$ and $rank(X) = p$. The math below will be different for $p > n$ and $rank(X) < p$ but the intuition is the same.

First we will explain that the principal component matrix can be obtained by singular value decomposition is used. Singular value decomposition means that the following holds for a matrix X:

$$X_{n\times p} = U_{n\times p} D_{p\times p} A'_{p\times p}$$

such that

- $U'U = I_p$,
- $A'A = AA' = I_p$ (hence A is an orthogonal matrix),
- $D = \begin{bmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & ... & 0 \\ 0 & 0 & 0 & d_p \end{bmatrix}$

Now it can be noted that if we multiply X as defined above and A we will get,

$$X_{n\times p} A_{p\times p} = UDA'A = UDA'A = UDI_p = UD.$$

This will turn out to be equal to $Z_{n\times p}$. To understand this, let's explain the properties of the loading of the principal components (which are imposed by PCA when computing the principal components and their loadings), which are:

$a'_i a_i = 1$ *for all* $i = 1, \ldots, p$ (norm of size 1)
$a'_i a_j = 0$ *for* $i \neq j$ (orthogonality)

If we expand $Z = XA$, we will get the following;

$$\begin{aligned} Z_{n\times p} &= \begin{bmatrix} Z_1 & ... & Z_p \end{bmatrix} \\ &= XA \\ &= X \begin{bmatrix} a_1 & ... & a_p \end{bmatrix} \\ &= \begin{bmatrix} Xa_1 & ... & Xa_p \end{bmatrix} \end{aligned}$$

where $a_1 \ldots a_p$ are the columns of A. It happens that $A'A$ satisfies the two constraints on the loadings. Therefore $Z = XA$ is the matrix of the principal components.

With the singular value decomposition, we can also show that the Xs are linear combinations of the principal components in the following way:

$$\begin{aligned} X &= UDA' \\ &= ZA' \\ \begin{bmatrix} X_1 & ... & X_p \end{bmatrix} &= \begin{bmatrix} Z_1 & ... & Z_p \end{bmatrix} \begin{bmatrix} a'_1 \\ \vdots \\ a'_p \end{bmatrix} \\ &= Z_1 a'_1 + Z_2 a'_2 + ... + Z_p a'_p \end{aligned}$$

Now if we write a_j as

$$a_j = \begin{bmatrix} a_{1j} \\ \vdots \\ a_{pj} \end{bmatrix}$$

we will see that each X_j is indeed a linear combination of the principal components

$$X_j = Z_1 a_{1j} + Z_2 a_{2j} + ... + Z_p a_{pj}$$

This shows that each X is a combination of all the principal components. Moreover it enables us to write:

$$X_j = Z_1 a_{1j} + \epsilon_j$$

This means that if we have the pre-estimator of Z_1 we can regress X_j on Z_1 and leave all the other variables in the error term. This does not lead to an omitted variable bias since all variables are uncorrelated with Z_1, since they are all orthogonal (as imposed by PCA). Therefore, we will be able to consistently estimate a_{1j}.

The NIPALS algorithm will indeed result in an (pre-)estimate of the $Z's$. How this is done will be explained next.

First of all, the NIPALS algorithms requires to standardize X. Initially the NIPALS algorithm begins by arbitrarily creating an initial column of nonzero numbers for $Z_1^{(I)}$. For example, it can be set equal to X_1. Then every column in X is regressed onto this initial column $Z_1^{(I)}$ in the following way:

$$a_{j1} = \frac{Z_1' X_j}{Z_1' Z_1}$$

Hence, this gives us an estimate of the loadings, $a_{j1}^{(1)}$. The loading vector $a_{j1}^{(I)}$ does not have unit length yet. So we rescale to a norm of 1 with the equation:

$$a_{j1}^{(II)'} = \frac{a_{j1}^{(I)}}{\sqrt{a_1^{(I)'} a_1^{(I)}}}$$

This procedure is repeated for each column in X until the entire vector a_j is filled. We then regress every row in X, denoted by rx_i, onto this normalized loading vector, which will give us a new value for Z, in the following way:

$$z_{i1}^{(II)} = \frac{rx_i' a_1^{(II)}}{a_1^{(II)'} a_1^{(II)}} = rx_i' a_1^{(II)}$$

We then repeat this step by substituting the new generated Z_1 in the regression of until we reach convergence. However, we still need estimates of the other Zs to be generated to calculated all the desired PC. For example, if we wish to obtain and (pre-)estimator for Z_2 we can use that $\epsilon = X - Z_1 a_1'$. Then we repeat the same steps we used to find Z_2 but we regress on ϵ instead of X. We then iterate the regressions until there is only a very small difference between iterations. Then we can move on to calculating the next PC if we wish. This means that the second - and other desired - PC are calculated on the residuals, ϵ, which are obtained after extracting the earlier PC. This also shows why each PC is orthogonal to the others. Each new PC is only seeing variation remaining after removing all the others. So there is no possibility that two PCs can explain the same type of variability.

What is left is explaining why NIPALS can consistently estimate the PC solution for **factor models**. The set-up of a factor model is as follows;

$$X_{ij} = \alpha_{i1} f_{j1} + \alpha_{i2} f_{j2} + ... + \alpha_{ir} f_{jr} + \epsilon_{ij} = \alpha_i' f_j + \epsilon_{ij}$$

where

$$\alpha_i' = \begin{bmatrix} \alpha_{i1} & ... & \alpha_{ir} \end{bmatrix}$$

$$f_j = \begin{bmatrix} f_{j1} \\ \dots \\ f_{jr} \end{bmatrix}$$

$$i = 1, \dots, n$$

$$j = 1, \dots, p$$

This is very similar to NIPALS. The NIPALS algorithm works because X is a linear combination of the principal components. Though we do not have to take all principal components, we can write X as a linear combination of those chosen principal components plus an error term. The similarity between the factor model (left side variables) and NIPALS (right side variables) is shown here:

$$\alpha = a$$
$$f = Z$$

In matrix form, the factor model is given with the equation:

$$X = FB' + \epsilon$$

where in the NIPALS algorithm we have that:

$$X = ZA'$$

Which is the same formula as the factor model, showing that the NIPALS algorithm is estimating the PC solution to a factor model.

3 Linear Discriminant Analysis

3.1

The posterior probability $p_1(x)$ using the Bayes rule is derived in the following way:

$$\begin{aligned}
\mathbb{P}[Y = 1 \mid X = x] &= \frac{\mathbb{P}[X = x \mid Y = 1] \cdot \mathbb{P}[Y = 1]}{\mathbb{P}[X = x]} \\
&= \frac{\pi_1 f_1(x)}{\pi_1 f_1(x) + \pi_2 f_2(x)} \\
&= \frac{\pi_1 \cdot \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(x-\mu_1)^2}{2\sigma^2}\right\}}{\pi_1 \cdot \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(x-\mu_1)^2}{2\sigma^2}\right\} + \pi_2 \cdot \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(x-\mu_2)^2}{2\sigma^2}\right\}} \\
&= \frac{\pi_1 \exp\left\{-\frac{(x-\mu_1)^2}{2\sigma^2}\right\}}{\pi_1 \exp\left\{-\frac{(x-\mu_1)^2}{2\sigma^2}\right\} + \pi_2 \exp\left\{-\frac{(x-\mu_2)^2}{2\sigma^2}\right\}}
\end{aligned}$$

By expanding the squares, we can see that this further breaks down into:

$$\begin{aligned}
&= \frac{\pi_1 \exp\left\{-\frac{x^2}{2\sigma^2}\right\} \cdot \exp\left\{-\frac{\mu_1^2}{2\sigma^2}\right\} \cdot \exp\left\{\frac{2\mu_1 x}{2\sigma^2}\right\}}{\pi_1 \exp\left\{-\frac{x^2}{2\sigma^2}\right\} \cdot \exp\left\{-\frac{\mu_1^2}{2\sigma^2}\right\} \cdot \exp\left\{\frac{2\mu_1 x}{2\sigma^2}\right\} + \pi_2 \exp\left\{-\frac{x^2}{2\sigma^2}\right\} \cdot \exp\left\{-\frac{\mu_2^2}{2\sigma^2}\right\} \cdot \exp\left\{\frac{2\mu_2 x}{2\sigma^2}\right\}} \\
&= \frac{\pi_1 \exp\left\{-\frac{\mu_1^2}{2\sigma^2}\right\} \cdot \exp\left\{\frac{2\mu_1 x}{2\sigma^2}\right\}}{\pi_1 \exp\left\{-\frac{\mu_1^2}{2\sigma^2}\right\} \cdot \exp\left\{\frac{2\mu_1 x}{2\sigma^2}\right\} + \pi_2 \exp\left\{-\frac{\mu_2^2}{2\sigma^2}\right\} \cdot \exp\left\{\frac{2\mu_2 x}{2\sigma^2}\right\}} \\
&= \frac{\pi_1 \exp\left\{\frac{2\mu_1 x - \mu_1^2}{2\sigma^2}\right\}}{\pi_1 \exp\left\{\frac{2\mu_1 x - \mu_1^2}{2\sigma^2}\right\} + \pi_2 \exp\left\{\frac{2\mu_2 x - \mu_2^2}{2\sigma^2}\right\}}
\end{aligned}$$

3.2

Using a), we want prove that $\log\left(\frac{p_1(x)}{1-p_1(x)}\right) = c_0 + c_1 x$ by deriving the formulae for the constants c_0 and c_1 as functions of μ_1, μ_2, σ^2, π_1, and π_2.

$$\begin{aligned}
\log\left(\frac{p_1(x)}{1-p_1(x)}\right) &= \log\left(\frac{p_1(x)}{p_2(x)}\right) \\
&= \log\left(\frac{\pi_1 \exp\left\{\frac{2\mu_1 x - \mu_1^2}{2\sigma^2}\right\}}{\pi_2 \exp\left\{\frac{2\mu_2 x - \mu_2^2}{2\sigma^2}\right\}}\right) \\
&= \log\left(\frac{\pi_1}{\pi_2}\right) + \log\left(\exp\left\{\frac{2\mu_1 x - \mu_1^2}{2\sigma^2} - \frac{2\mu_2 x - \mu_2^2}{2\sigma^2}\right\}\right) \\
&= \log\left(\frac{\pi_1}{\pi_2}\right) + \frac{2\mu_1 x - \mu_1^2 - 2\mu_2 x + \mu_2^2}{2\sigma^2} \\
&= \log\left(\frac{\pi_1}{\pi_2}\right) + \frac{2(\mu_1 - \mu_2)x + \mu_2^2 - \mu_1^2}{2\sigma^2} \\
&= \log\left(\frac{\pi_1}{\pi_2}\right) + \frac{\mu_2^2 - \mu_1^2}{2\sigma^2} + \frac{\mu_1 - \mu_2}{\sigma^2} x,
\end{aligned}$$

so we have that $\log\left(\frac{p_1(x)}{1-p_1(x)}\right) = c_0 + c_1 x$ for $c_0 = \log\left(\frac{\pi_1}{\pi_2}\right) + \frac{\mu_2^2 - \mu_1^2}{2\sigma^2}$ and $c_1 = \frac{\mu_1 - \mu_2}{\sigma^2}$. Note that c_0 and c_1 are constants. □

3.3

The derivation in 3.2 tells us about the relationship between LDA and logistic regression. In logistic regression, we find the log-odds ratio $\frac{p_1(x)}{1-p_1(x)}$ to equal a form alike $\beta_0 + \beta_1 x$, where β_0 and β_1 are constants (coefficients). As such, LDA gives a similar linear functional form in x with coefficients $\beta_0 = \log\left(\frac{\pi_1}{\pi_2}\right) + \frac{\mu_2^2 - \mu_1^2}{2\sigma^2}$ and of $\beta_1 = \frac{\mu_1 - \mu_2}{\sigma^2}$. Therefore, from a logistic regression, we directly get the

estimated coefficients for β_0 and β_1 while for LDA we (might) need to estimate more parameters, in this case we need to estimate μ_1, μ_2, and σ^2 (π_1 and π_2 are prior probabilities based on our beliefs and would be taken as given). When considering only the number of parameters to estimate, one may prefer to run a logistic regression instead of LDA. However, if one has strong prior beliefs about the class probabilities, LDA may seem more applicable due to incorporating Bayes rule.

3.4

We want to show that the Bayes classifier assigns an observation $X = x$ to class 1 if $\delta_1(x) > \delta_2(x)$, where $\delta_k(x) = x'\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k'\Sigma^{-1}\mu_k + \log(\pi_k)$ for $k = 1, 2$.

Note that the Bayes classifier assigns an observation $X = x$ to the class k for which the posterior probability $\mathbb{P}[Y = k \mid X = x]$ is the largest, that is when:

$$\frac{\pi_1 f_1(x)}{\pi_1 f_1(x) + \pi_2 f_2(x)} > \frac{\pi_2 f_2(x)}{\pi_1 f_1(x) + \pi_2 f_2(x)} \iff \pi_1 f_1(x) > \pi_2 f_2(x)$$

since the denominator $\pi_1 f_1(x) + \pi_2 f_2(x) > 0$. Consider the following derivation:

$$\begin{aligned}
& \pi_1 f_1(x) > \pi_2 f_2(x) \\
\iff & \frac{\pi_1}{(2\pi)^{p/2}|\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(x-\mu_1)'\Sigma^{-1}(x-\mu_1)\right\} > \frac{\pi_2}{(2\pi)^{p/2}|\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(x-\mu_2)'\Sigma^{-1}(x-\mu_2)\right\} \\
\iff & \pi_1 \exp\left\{-\frac{1}{2}(x-\mu_1)'\Sigma^{-1}(x-\mu_1)\right\} > \pi_2 \exp\left\{-\frac{1}{2}(x-\mu_2)'\Sigma^{-1}(x-\mu_2)\right\}.
\end{aligned}$$

Since the logarithm is an increasing function, we can apply it to both sides of the inequality and retain the sign:

$$\begin{aligned}
\iff & \log\left(\pi_1 \exp\left\{-\frac{1}{2}(x-\mu_1)'\Sigma^{-1}(x-\mu_1)\right\}\right) > \log\left(\pi_2 \exp\left\{-\frac{1}{2}(x-\mu_2)'\Sigma^{-1}(x-\mu_2)\right\}\right) \\
\iff & \log(\pi_1) - \frac{1}{2}(x-\mu_1)'\Sigma^{-1}(x-\mu_1) > \log(\pi_2) - \frac{1}{2}(x-\mu_2)'\Sigma^{-1}(x-\mu_2) \\
\iff & \log(\pi_1) - \frac{1}{2}\left(x'\Sigma^{-1}x - 2x'\Sigma^{-1}\mu_1 + \mu_1'\Sigma^{-1}\mu_1\right) > \log(\pi_2) - \frac{1}{2}\left(x'\Sigma^{-1}x - 2x'\Sigma^{-1}\mu_2 + \mu_1'\Sigma^{-1}\mu_2\right)
\end{aligned}$$

Note that $-\frac{1}{2}x'\Sigma^{-1}x$ appears on both sides of the inequality since it is independent of the class. As such, it drops out. By working out the brackets we get:

$$\begin{aligned}
\iff & \log(\pi_1) + x'\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_1'\Sigma^{-1}\mu_1 > \log(\pi_2) + x'\Sigma^{-1}\mu_2 - \frac{1}{2}\mu_2'\Sigma^{-1}\mu_2 \\
\iff & \delta_1(x) > \delta_2(x)
\end{aligned}$$

□